UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Convolutional Neural Networks for food recognition

Trabajo de Final de Grado

Presentado a la

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

Por

Alberto Palacio Franco

En cumplimiento parcial de los requisitos para el grado de Ingeniería en Sistemas Audiovisuales

Tutor: José Adrián Rodríguez Fonollosa

Barcelona, Enero 2017

Abstract

This project consists on an application made in Matlab, where different foods are recognized through a convolutional neural network, then stored it in a SQL database with its quantity to create "your fridge". Thanks to this you can know what aliments you have. Finally, these aliments are compared with other database that contains many recipes to find out which can be done.

Resum

El projecte consisteix en una aplicació realitzada en Matlab. Consisteix en el reconeixement de diferents aliments en imatges, a través d'una xarxa neuronal, seguidament s'emmagatzemen juntament amb una quantitat en una base de dades SQL per crear "tu nevera" i saber els aliments dels que disposes perquè finalment, es comparin aquests aliments amb una altra base de dades amb receptes i així mostrar les que pots fer amb els aliments que disposes.

Resumen

El proyecto consiste en una aplicación realizada en Matlab. Consiste en el reconocimiento de diferentes alimentos en imágenes, a través de una red neuronal, seguidamente se almacenan junto con una cantidad en una base de datos SQL para crear “tú nevera” y saber los alimentos de los que dispones para que finalmente, se comparen estos alimentos con otra base de datos con recetas y así mostrar las que puedes realizar con los alimentos que dispones.

Agradecimientos

A mis padres y hermanos, por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante.

A mi novia, por todo, porque sin ella este proyecto no habría sido posible.

A mis amigos, por dedicar parte de su tiempo en ayudarme a obtener gran cantidad de las fotos que he necesitado

A mi tutor, por toda la paciencia y tiempo gastado a lo largo de estos meses para ayudarme en la elaboración de este trabajo.

Historial de revisiones y registro de aprobación

Revisión	Fecha	Propósito
0	14/11/2016	Creación del documento
1	06/01/2017	Revisión del documento

DOCUMENT DISTRIBUTION LIST

Nombre	e-mail
Alberto Palacio Franco	albertopf@gmail.com
José Adrián Rodríguez Fonollosa	jose.fonollosa@upc.edu

Escrito por:		Revisado y aprobado por:	
Date	14/11/2016	Date	06/01/2017
Name	Alberto Palacio Franco	Name	José Adrián Rodríguez Fonollosa
Position	Autor del proyecto	Position	Supervisor del proyecto

Tabla de contenidos

Listado de Figuras

Listado de Tablas:

1. Introducción

La idea de este proyecto surgió tras repetirse la situación de llegar a casa después de la jornada laboral y preguntarse ¿Qué puedo hacerme para cenar? Estoy cansado, quiero algo rápido de hacer. Muchas personas no saben con exactitud de que alimentos disponen en su nevera y que recetas rápidas pueden hacer con ellos. Conociendo el problema, surgió la solución, una aplicación en la que poder almacenar todos los alimentos de los que disponemos de una manera tan sencilla como haciéndole una foto, pero con ello no solucionábamos la parte de que cocinar, por lo que esa aplicación también tenía que disponer de una base de datos con distintas recetas rápidas y sencillas, que poder realizar con los alimentos de los que disponemos en nuestra nevera. Y así nació la aplicación Fast Cooking.

1.1. Estructura del Proyecto

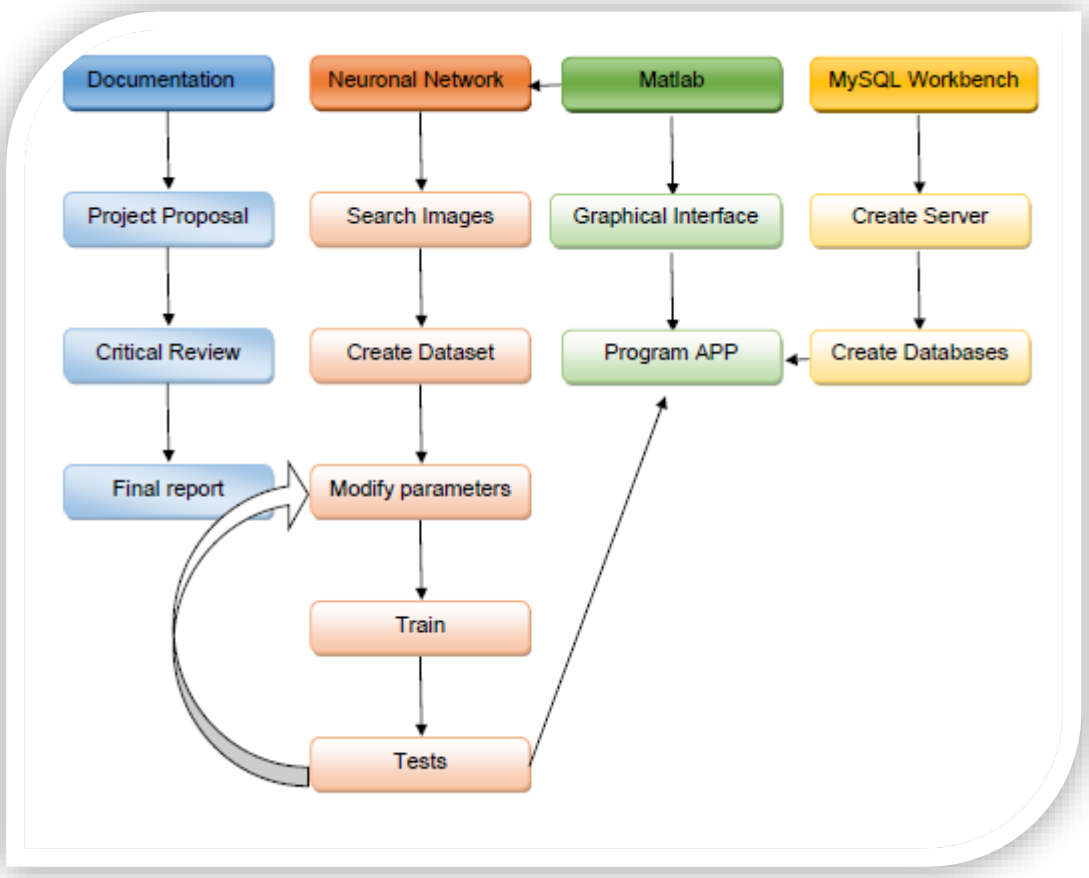


Figura 1: Estructura del Proyecto mediante un diagrama de árbol

En la figura podemos observar los bloques de trabajo en los que se ha dividido el proyecto, así como los pasos a seguir en cada uno de ellos:

- Documentación: En este bloque se encuentran los tres documentos a realizar a lo largo del presente trabajo.

- Matlab: Los trabajos que se han realizado en este apartado son la interfaz gráfica y la programación de la aplicación. Esta se interconecta con los dos bloques restantes.
- Red Neuronal: Este es el bloque más amplio del trabajo, en él se lleva a cabo desde el proceso de búsqueda y creación del "dataset" de imágenes de alimentos hasta el entrenamiento y validación en la red. Se encuentra conectado con el bloque de Matlab, ya que la herramienta que se ha utilizado necesita ser implementada en este software para su funcionamiento.
- MySQL Workbench: El último bloque que podemos observar es el encargado de la creación del servidor y del trabajo con el lenguaje SQL, se ha creado en este apartado un servidor local y las tablas SQL con la base de datos de alimentos y recetas que se alojarán en él para comunicarlo finalmente con Matlab para su utilización.

1.2. Objetivos

Los objetivos principales del proyecto son:

- Conseguir un reconocimiento satisfactorio de los alimentos.
- Funcionamiento eficiente y sin errores de la aplicación.
- Crear una aplicación innovadora para una futura implementación en Smartphone.

Considerando estos objetivos, y con el tiempo planificado para el proyecto, las bases de datos, tanto de alimentos como de recetas, se ha relegado su tamaño a un segundo plano debido a la ausencia de necesidad de conocimientos técnicos para su realización. Por todo ello, se han escogido un número reducido de ambos, los cuales se han considerado los mínimos para poder realizar las pruebas correctamente.

1.3. Requerimientos y Especificaciones

Requerimientos del proyecto:

- Matlab R2015a [1]
- MySQL Workbench 6.3 [2]
- Appserver [3]
- ODCB server [4]
- Conocimientos de programación en Matlab y SQL
- Base de datos de imágenes de alimentos
- Base de datos de recetas

Especificaciones del proyecto:

Error	TOP1	TOP5
ENTRENAMIENTO	<3,5%	<1%
VALIDACIÓN	<35%	<10%

Tabla 1: Rango de errores aceptables en el entrenamiento de la red neuronal.

1.4. Métodos y procedimientos

Este es un proyecto nuevo e independiente creado a partir de una idea propia. Para llevarlo a cabo se han utilizado los conocimientos del autor en Matlab y SQL, además de la ayuda de software externo descrito a continuación:

- MatCovNet, es una herramienta para implementar una red neuronal en Matlab [5]
- Programa de DataAugmentation realizado por Takuya Minagawa [6]

1.5. Plan de trabajo

<table>
<tr><td>Project: Matlab</td><td colspan="2">WP ref: 1</td></tr>
<tr><td>Major constituent: Software</td><td colspan="2">Sheet 1 of 5</td></tr>
<tr><td rowspan="2">Short description:
Application done in Matlab</td><td colspan="2">Planned start date: 11/02/16
Planned end date: 10/11/16</td></tr>
<tr><td colspan="2">Start event: 11/02/16
End event:</td></tr>
<tr><td>Internal task T1: Graphical Interface
Internal task T2: Program APP</td><td>Deliverables:
T1
T2</td><td>Dates:
31/05/16
31/08/16</td></tr>
</table>

<table>
<tr><td>Project: Neuronal Network</td><td colspan="2">WP ref: 2</td></tr>
<tr><td>Major constituent: Software</td><td colspan="2">Sheet 2 of 5</td></tr>
<tr><td rowspan="2">Short description:
Use a Neuronal Network to recognize aliments</td><td colspan="2">Planned start date: 11/02/16
Planned end date: 10/11/16</td></tr>
<tr><td colspan="2">Start event: 11/02/16
End event:</td></tr>
<tr><td>Internal task T1: Learn about CNN
Internal task T2: Search Images
Internal task T3: Search Recipes
Internal task T4: Create Dataset
Internal task T5: Modify Parameters
Internal task T6: Train
Internal task T7: Test</td><td>Deliverables:
T4
T7</td><td>Dates:
15/09/16
10/11/16</td></tr>
</table>

<table>
<tr><td>Project: MySQL Workbench</td><td colspan="2">WP ref: 3</td></tr>
<tr><td>Major constituent: Software</td><td colspan="2">Sheet 3 of 5</td></tr>
<tr><td rowspan="2">Short description:
Use to create server and work with SQL</td><td colspan="2">Planned start date:11/07/16
Planned end date:04/08/16</td></tr>
<tr><td colspan="2">Start event:11/07/16
End event:</td></tr>
<tr><td>Internal task T1: Learn about localhost servers
Internal task T2: Create Server
Internal task T3: Learn SQL
Internal task T4: Create Databases</td><td>Deliverables:
T4</td><td>Dates:
4/08/16</td></tr>
</table>

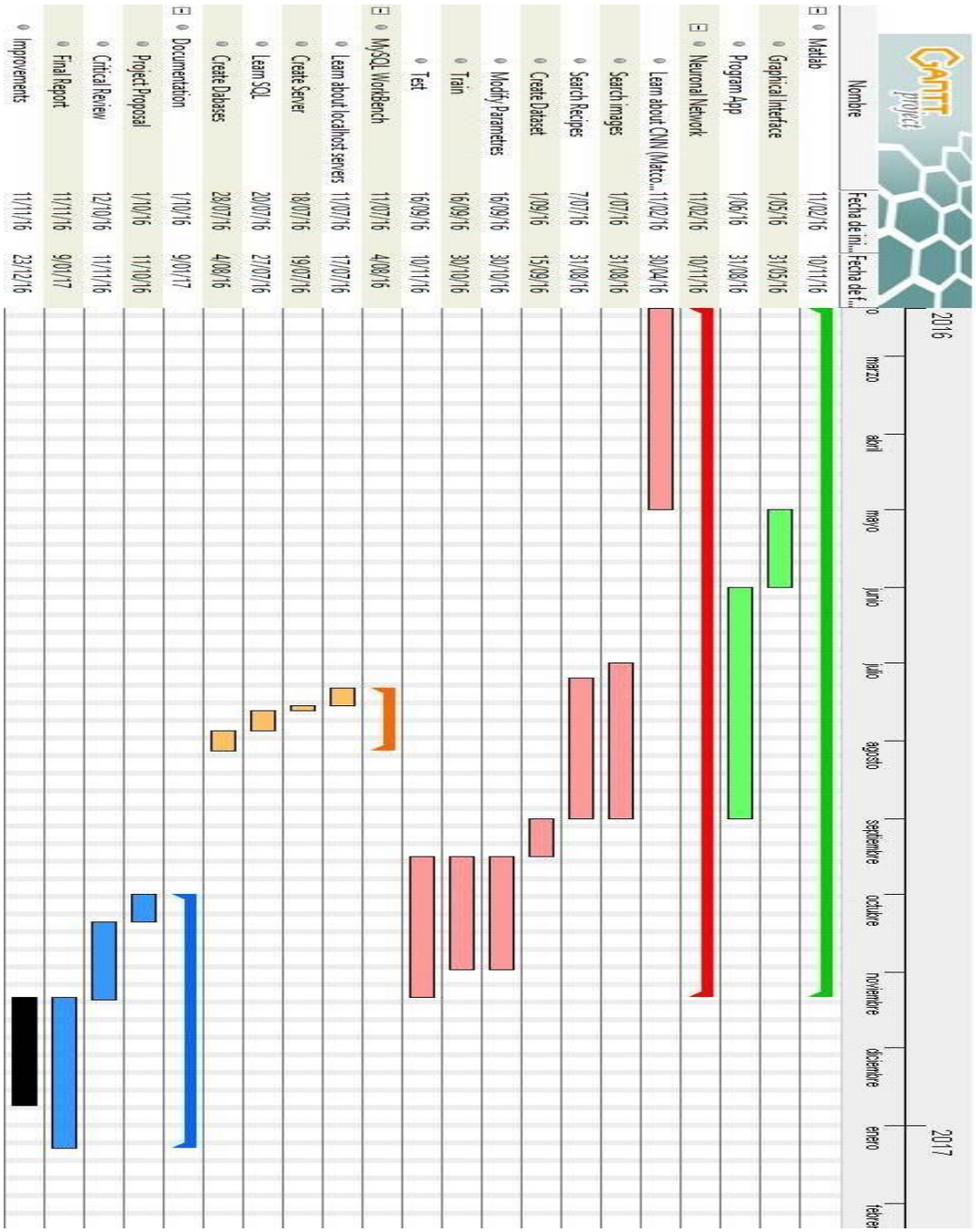
<table>
<tr><td>Project: Documentation</td><td colspan="2">WP ref: 4</td></tr>
<tr><td>Major constituent: Documents</td><td colspan="2">Sheet 4 of 5</td></tr>
<tr><td rowspan="2">Short description:
Documents to deliver</td><td colspan="2">Planned start date: 01/10/16
Planned end date: 9/01/17</td></tr>
<tr><td colspan="2">Start event:9/01/17
End event:</td></tr>
<tr><td>Internal task T1: Project Proposal
Internal task T2: Critical Review
Internal task T2: Final Report</td><td>Deliverables:
T1
T2
T3</td><td>Dates:
11/10/16
11/11/16
09/01/17</td></tr>
</table>

Hitos

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1	Graphical Interface	Graphical Interface	29
1	2	Program APP	Program APP	16
2	1	Learn about CNN	Learn about CNN	11
2	2	Search Images	Search Images	29
2	3	Search Recipes	Search Recipes	29
2	4	Create Dataset	Create Dataset	31
2	5	Modify Parameters	Modify Parameters	37
2	6	Train	Train	37
2	7	Test	Test	39
3	1	Learn about servers	Learn about localhost servers	22
3	2	Create Server	Create Server	23
3	3	Learn SQL	Learn SQL	24
3	4	Create Databases	Create Databases	25

4	1	Project Proposal	Project Proposal	34
4	2	Critical Review	Critical Review	39
4	3	Final Report	Final Report	47

1.6. Diagrama de Gantt



2. Conocimientos del entorno:

En este apartado se explican los conocimientos previos necesarios para entender el proyecto que se ha llevado a cabo. Se ha dividido en cuatro bloques.

2.1. Matlab

Se trata de un software matemático que dispone de un lenguaje propio y está disponible para los entornos Windows, Mac OS, Unix y Linux. Gracias a este programa se pueden manipular todo tipo de elementos matemáticos, así como la implementación de algoritmos, creación de interfaces gráficas y dispone de la posibilidad de llamar funciones y subrutinas escritas en C (MEX-files), esto último se ha necesitado en el presente trabajo para la implementación de la herramienta MatConvNet. La versión utilizada en este proyecto es la Matlab R2015a.

2.2. Base de datos

Para realizar las bases de datos se necesitan conocimientos en SQL (Structured Query Languaje) [7], se trata de un lenguaje de programación con el que se pueden gestionar diversas tablas. Dispone de diversos comandos para poder seleccionar, crear, modificar, insertar o borrar tablas. Adicionalmente, al utilizar estos comandos se pueden complementar con filtros como ordenación o selección. Para este proyecto se han adquirido los conocimientos necesarios para el tratamiento de tablas para su aplicación en las bases de datos que se han necesitado.

Por otra parte se ha necesitado una herramienta para la configuración de servidores, el software que se ha utilizado es el “MySQL Workbench 6.3” [2], gracias a este software se puede gestionar un servidor y ejecutar scripts SQL.

Para finalizar, se han necesitado los conocimientos necesarios en Matlab para poder conectarse al servidor y poder gestionarlo desde él [8].

2.3. Redes Neuronales Convolucionales(CNN)

Una red de neuronas es una herramienta matemática que modela, de forma muy simplificada, el funcionamiento de las neuronas en el cerebro.

En el caso de este proyecto sería una red de neuronas que detecte alimentos en imágenes. Esta red recibiría tantos números a su entrada como píxeles tienen nuestras imágenes (o tres por cada píxel si utilizamos imágenes en color). Y si la información que esperamos a la salida es que nos diga de qué alimento se trata. A la salida obtendremos tantos números como alimentos a entrenar, podemos imaginar que si el número que sale de la red toma un valor cercano a 1.0, significa que se trata del alimento más probable, conforme se acerca a 0.0, disminuye su probabilidad.

En el siguiente diagrama podemos ver la arquitectura de una red de neuronas. Cada círculo representa una neurona. Las neuronas se organizan en capas, de la siguiente forma: las neuronas amarillas son las entradas, y reciben cada uno de los números de nuestra lista de números entrante, las neuronas verdes son las salidas, y una vez que la red realiza su operación matemática, contienen el resultado, también

como una lista de números; las neuronas grises son neuronas ocultas, que contienen cálculos intermedios de la red.

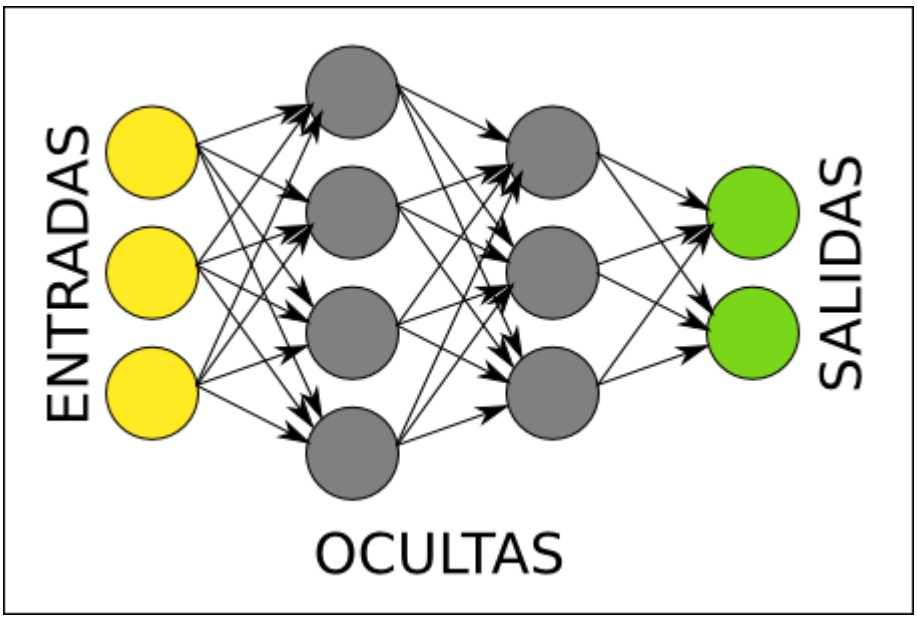


Figura 2: Diagrama del funcionamiento de una Red Neuronal

Normalmente todas las neuronas de cada capa tienen una conexión con cada neurona de la siguiente capa, como se representa en el diagrama. Estas conexiones tienen asociado un número, que se llama peso. La principal operación que realiza la red de neuronas consiste en multiplicar los valores de una neurona por los pesos de sus conexiones salientes. Cada neurona de la siguiente capa recibe números de varias conexiones entrantes, y lo primero que hace es sumarlos todos.

En este trabajo se ha llevado a cabo el entrenamiento gracias a una herramienta diseñada para Matlab, MatConvNet [5], que consiste en una serie de funciones de Matlab que ejecutan los bloques de construcción de la CNN. Se realiza mediante la función “vl_simplenn”, ya que este modelo es adecuado para redes que tienen una topología lineal, es decir, una cadena de bloques de cálculo, como es nuestro caso.

2.4. Data Augmentation

Se trata de un software que se encarga de multiplicar la cantidad de imágenes que disponemos para nuestro entrenamiento de la red neuronal, cambiando el ratio de aspecto, difuminando, rotando añadiendo ruido a las fotos originales. Con este sencillo programa podemos llegar a tener un gran “dataset” sin necesidad de realizar todas las capturas manualmente. Se ha conseguido una aplicación sencilla e intuitiva para nuestro proyecto [6]. Este software solo está disponible para el entorno Windows, pero su rendimiento ha sido satisfactorio, se eligen los rangos de variación de los parámetros así como los directorios de entrada y salida. En nuestro caso se ha conseguido pasar de alrededor de 400 a 60.000 imágenes.

3. Metodología / Desarrollo del proyecto:

El proyecto se divide en tres grandes bloques que se explican a continuación, el código de todas las funciones creadas para este proyecto se puede ver en el apéndice 5:

- Matlab, en el que se realiza la interfaz gráfica y la aplicación.
- Servidor MySQL, en el que crearemos las dos bases de datos necesarias, "mi nevera" y "recetas", se enlaza directamente con la aplicación.
- Red Neuronal, este bloque es el más costoso de los tres, en él se buscarán fotos de distintos alimentos desde diferentes dispositivos de imagen. Posteriormente gracias a un programa de "Data Augmentation", se aumentará la cantidad hasta conseguir el número deseado para el correcto entrenamiento. La última parte será el entrenamiento de la red neuronal gracias y su implementación en Matlab.

3.1. Matlab

En este apartado se presenta la interfaz gráfica, así como el funcionamiento general de la aplicación.

Interfaz gráfica - Aplicación

Se buscaba una interfaz simple e intuitiva por lo que realizarla en Matlab, se creyó la opción más apropiada. Tanto la aplicación como la interfaz se dividen en cuatro apartados:

- Menú Principal: Se trata del índice de nuestra aplicación, desde donde podemos acceder a cada uno de los apartados de la aplicación.

Figura 3: Menú Principal

- Guardar Alimentos: Se trata del apartado de la aplicación donde implementaremos la red neuronal, parte que trataremos posteriormente. En base a lo que la aplicación se refiere, podemos ver en la figura siguiente la distribución de las distintas funciones.
    a. *Seleccionar foto:* Se abre el explorador para buscar la imagen del alimento que deseamos añadir a "Mi Nevera"
    b. *Alimento:* En esta caja aparecerá el nombre del ingrediente una vez hayamos seleccionado la foto. Existe la opción de ingresar el nombre manualmente en caso de que no se quiera seleccionar una foto o de que se haya identificado erróneamente.
    c. Cantidad: Aquí se ingresa manualmente la cantidad del ingrediente que disponemos.
    d. *Guardar:* Con este botón se procede a añadir a la base de datos "mi nevera" tanto el alimento como la cantidad. Primero, se carga el listado de ingredientes, así como la base de datos "Mi Nevera". Seguidamente se comprueba si el ingrediente se encuentra en la lista, en caso de que no aparezca, se añade. Por último se busca en la base de datos y en caso de que el alimento ya se encuentre añadido, se sumará la cantidad a la que tiene, en caso contrario, se ingresará en "Mi Nevera" el nuevo ingrediente junto a su cantidad.
    e. *Limpiar:* Se borran las cajas "Alimento" y "Cantidad".
    f. Enlaces: Los tres botones inferiores nos direccionan al resto de apartados. Se encuentran en todas las ventanas de la aplicación.

Figura 4: Guardar Alimentos

- Mi Nevera: Al acceder a esta sección, se carga la base de datos “Mi Nevera” y se carga en el “listbox”, allí aparecen los alimentos junto a la cantidad que disponemos. Esta ventana sirve para tres opciones:
    a. Comprobar los alimentos que disponemos y sus cantidades.
    b. Si seleccionamos uno de los ingredientes y presionamos el botón “Quitar”, la cantidad pasará a ser “0”.
    c. En el caso de apretar el botón “Modificar”, la cantidad del ingrediente seleccionado se cambiará por el valor que se ha introducido previamente en la caja. Modificar la cantidad a “0” cumple la misma función que el botón “Quitar”.

  Al dejar la cantidad de un alimento a “0”, este no desaparecerá del listado, pese a no contar como ingrediente disponible para el siguiente apartado. Esto es debido a que así puedes darte cuenta de los alimentos que te faltan y debes comprar.

Figura 5: Mi Nevera

- Recetas: Al acceder a este último apartado, se carga la base de datos de recetas y se comparan los ingredientes que tenemos en “Mi Nevera” para identificar cuáles son las recetas que podemos realizar con los alimentos que disponemos. Por todo ello, al accionar el “pop up”, aparecerán en el desplegable todas las recetas disponibles.

Figura 6: Recetas

Si seleccionamos una de ellas aparecerá una foto de la receta, así como la descripción de ella (tiempo, ingredientes necesarios, pasos a seguir), a continuación se puede ver un ejemplo.

Figura 7: Ejemplo Receta

3.2. Servidor MySQL

Lo primero es instalar el programa "AppServ" para convertir nuestro ordenador en un servidor. Seguidamente se instalan los drivers necesarios, en nuestro caso al ser un Windows 8.1, se debe instalar el "ODCB server". Finalmente se instala el programa "MySQL Workbench". Una vez abierto el programa se accede al servidor local.

Mi Nevera/Recetas

Se ha creado un script SQL en el mismo programa (Ver Apéndice1), donde se encuentra toda la información inicial de las bases de datos. Al ejecutar este script se crea las cuatro tablas siguientes:

- ingredients: En esta tabla se muestran los ingredientes conocidos de la aplicación, se pueden ver en el Apéndice2, inicialmente se han añadido 15 para realizar pruebas, 10 de ellos están entrenados en la red neuronal y los 5 restantes para pruebas manuales. Cada alimento insertado en la aplicación que no aparezca en la tabla, se añade al final de ella.
- recipes: En esta tabla se muestran las recetas de la base de datos "Recetas", a la cual se accede desde la aplicación, se pueden ver en el Apéndice3.
- ing_fridge: En esta tabla se muestran los identificadores de los ingredientes y las cantidades de la base de datos "Recetas", desde la aplicación se accede a esta tabla y se modifican los identificadores por los nombres a través de la tabla "ingredients".
- ing_rec: Esta tabla sirve para añadir los ingredientes necesarios a cada receta, para así poder compararlo más eficazmente con la nevera a la hora de identificar los alimentos disponibles.

3.3. Red Neuronal

En este último apartado se explicará el trabajo realizado respecto al uso de una red neuronal en este proyecto. Se comenzará exponiendo los pasos seguidos para la obtención de la base de datos de imágenes, posteriormente se explicarán los métodos usados para la instalación de la herramienta MatConvNet [5] en Matlab, los procedimientos realizados en el entrenamiento y la implementación en la aplicación.

Imágenes/Data Augmentation

Primeramente se consiguieron fotos de 10 alimentos diferentes, se hicieron con diferentes dispositivos móviles (ver Apéndice 4) para conseguir distintas calidades de imagen y en entornos diferentes. Estas son las denominadas imágenes originales, se obtuvieron alrededor de 30-50 por alimento.

A continuación, se multiplican estas fotos a través del programa de aumento de datos que hemos conseguido [6], gracias a este software se generaran 6.000 imágenes, cambiando el ratio de aspecto, difuminando, rotando y añadiendo ruido a las originales. Previamente, estas se han dividido en

imágenes de entrenamiento y de test, y del total, 5.000 corresponden al primer grupo y el resto al segundo.

Se han utilizado cuatro scripts de matlab en este apartado:

- resize_pre: Previo a la multiplicación de imágenes, estas se normalizan en tamaño a 128x128 para una mayor eficiencia del uso del programa.
- resize_post: Con este script, modificaremos las 6.000 fotos de cada alimento a un nuevo tamaño, 32x32, el cual es el valor considerado óptimo para nuestra base de datos.
- resize_post_norm: Posteriormente, se normalizan las imágenes sustrayéndoles su media para su correcto entrenamiento.
- Save_dataset_x: Se crearon dos scripts, uno de “test” y otro de “train”. Con este código se leen las 60 .000 imágenes totales, junto con sus “labels” (etiqueta con el ingrediente al que pertenece la imagen) y se guardan en seis bloques iguales con el formato adecuado para el entrenamiento.

Con el dataset creado, se continúa instalando la herramienta MatConvNet y procediendo al entrenamiento de la red neuronal.

Entrenamiento/Implementación

Lo primero es conseguir la herramienta [5] para proceder a su instalación. A continuación se explicarán las funciones que necesitamos:

- vl_setupnn: Instala la herramienta MatConvNet. Sólo se ejecuta la primera vez.
- vl_compilenn: Compila los archivos MEX de la herramienta. Se ejecuta antes de abrir la aplicación.
- cnn_cifar: Al ejecutar ese script comienza el entrenamiento. Aquí se especifican las rutas del dataset y llama a la función de entrenamiento “cnn_train” además de a la función “cnn_cifar_init”, donde están definidos los parámetros y los bloques del entrenamiento. Cifar es un ejemplo propuesto por la misma herramienta, se ha utilizado este ya que se asemejaba a las características del proyecto, por lo que se consideró que podría ser de ayuda y solo se necesitaría modificar los parámetros de entrenamiento.
- extract_features: Script que se ha creado para enlazar el modelo con la aplicación, en él se carga el modelo que hemos entrenado, así como la imagen a reconocer. Esta se normaliza igual que las de la base de datos que hemos explicado en el apartado anterior. Se evalúa gracias a la función “vl_simplenn”, nos devuelve un conjunto de matrices, en la última de ellas, se escoge la posición del valor más alto y esta se corresponde al identificador del ingrediente de la imagen a reconocer. Finalmente, se identifica el alimento, y se envía a la aplicación para que aparezca en la celda que le corresponde.
- vl_simplenn: Función que evalúa la imagen seleccionada con el modelo entrenado.

Para la implementación en el código de la aplicación, se necesitará únicamente el script “extract features” que se ha explicado anteriormente. Al seleccionar la imagen que deseamos identificar, se guarda en una variable que se pasa a esta función y a su salida se obtiene el nombre del ingrediente en una cadena de caracteres. Para finalmente poder asignar este string a la celda correspondiente.

4. Resultados

En esta sección se explicaran los resultados obtenidos. Se han realizado alrededor de 20 entrenamientos con distintos parámetros. Consiguiendo a lo largo de todas estas pruebas una mejoría de un 10%, en las últimas 4, se ha mantenido el resultado invariable por lo que se ha asumido que se ha encontrado el resultado óptimo. A continuación se expone el entrenamiento realizado con mejores resultados.

Nºdrop outs	Valor drop out	Learning rate	Batch size	Nº epochs
1	0,3	1	128	45

Tabla 2: Valores de los parámetros usados en el entrenamiento

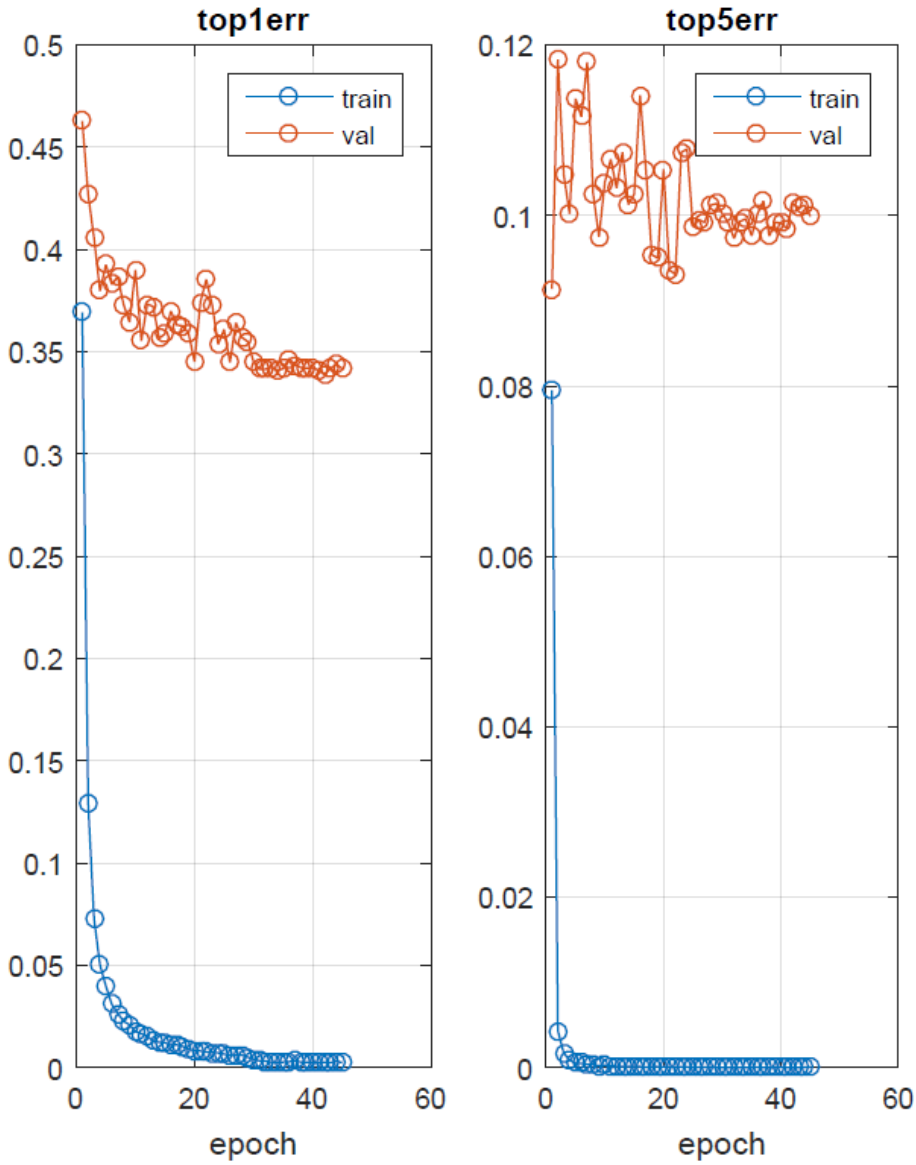


Figura 8: Errores de entrenamiento y validación.

Error	TOP1	TOP5
ENTRENAMIENTO	1%	0%
VALIDACIÓN	34%	10%

Tabla 3: Porcentaje de errores obtenidos

Figura 9: Identificación satisfactoria en la aplicación

Hemos podido ver en las tablas anteriores la configuración de los parámetros para los resultados obtenidos, así como el porcentaje de error en el entrenamiento para cada época. Se ha identificado la última, la 45, como óptima. El entrenamiento obtiene valores cercanos al cero, esto quiere decir que el entrenamiento se ha realizado correctamente. Los valores de la prueba de validación son más altos, pero dentro del rango. El valor top1 corresponde a los aciertos exactos del alimento, en el caso top5, significa que el alimento se encuentra entre los cinco más probables.

Finalmente podemos apreciar una figura en la que se muestra la aplicación en funcionamiento, en este caso se ha seleccionado una foto de una patata, y la ha reconocido satisfactoriamente.

5. Conclusiones y futuros desarrollos:

Al comienzo de este proyecto se consideraron tres objetivos principales y ahora se va a estudiar si se ha logrado llegar a un resultado satisfactorio. En el primero de ellos, los resultados que se han obtenido se encuentran dentro del rango de error mostrados en las especificaciones iniciales. Este rango se ha considerado tan amplio debido a la base de datos de las imágenes. Se utilizan 5.000 fotos por alimento para el entrenamiento, pero realmente son entre 20 y 40 las originales y aunque al gran trabajo que realiza el programa de aumento, son demasiadas las multiplicaciones que debe realizar. Todo ello conlleva a que haya poca variedad real en la totalidad de las imágenes y esto al aumento del error. Pese a todo lo comenta, se consideran los resultados óptimos para la base de datos de la que se dispone. Respecto al segundo de los objetivos, se han realizado pruebas de test de la aplicación para buscar posibles "bugs" en la aplicación, después de múltiples pruebas realizadas, no se ha encontrado ningún error o fallo en el funcionamiento de la aplicación creada. El tercer y último objetivo trataba de la parte innovadora de la aplicación, nos encontramos ante una aplicación única en el mundo de la telefonía móvil, ya que se han encontrado aplicaciones con recetas para realizar, pero en ninguna de ellas se pueden almacenar de los alimentos de los que dispones para poder filtrar los resultados automáticamente. Por todo ello se puede concluir, que los objetivos se han cumplido satisfactoriamente en el presente trabajo.

Una vez cumplidos los objetivos, se va a estudiar su posible futura implementación en dispositivos móviles. Aunque hay tres puntos importantes a desarrollar previamente:

- Conseguir una mayor cantidad de fotos originales para obtener una gran reducción del error de validación.
- Aumentar tanto el número de ingredientes a entrenar, como la cantidad de recetas.
- Utilizar un servidor que no sea local para optimizar la aplicación.

Adicionalmente, podrían añadir nuevas funcionalidades de la aplicación, como alergias, lista de la compra o calorías consumidas.

Por lo que siguiendo estos pasos, en un futuro podríamos ver la aplicación que se ha desarrollado, "Fast Cooking", totalmente operativa en nuestros smartphones.

Referencias

[1] «Matlab,» [En línea]. Available: https://es.mathworks.com/.

[2] «MySQL Workbench,» [En línea]. Available: http://www.mysql.com/products/workbench/.

[3] «AppServ,» [En línea]. Available: https://www.appserv.org/en/.

[4] «ODBC Server,» [En línea]. Available: https://www.microsoft.com/es-ES/download/details.aspx?id=36434.

[5] MatConvNet. [En línea]. Available: http://www.vlfeat.org/matconvnet/.

[6] T. Minagawa. [En línea]. Available: https://github.com/takmin/DataAugmentation.

[7] «SQL,» [En línea]. Available: https://es.wikipedia.org/wiki/SQL.

[8] «Database Toolbox Matlab,» [En línea]. Available: http://es.mathworks.com/help/database/.

[9] «Revuelto de tomate,» [En línea]. Available: http://recetasdecocina.elmundo.es/2015/10/revuelto-tomate-receta-casera.html.

[10] «Revuelto de tomate,» [En línea]. Available: http://olorahierbabuena.com/2015/10/huevos-revueltos-con-tomate.html.

[11] «Pulpo a la gallega,» [En línea]. Available: https://cookpad.com/es/recetas/270128-pulpo-a-la-gallega-rapido-y-facil.

[12] «Tortilla de patatas,» [En línea]. Available: http://www.cocinillas.es/2014/12/tortilla-de-patatas-rapida-y-sin-grasas/#jp-carousel-33403.

[13] «Plátanos fritos,» [En línea]. Available: https://cookpad.com/es/recetas/1449301-platanos-fritos.

[14] «Pimientos rellenos de crema de atún,» [En línea]. Available: https://cookpad.com/es/recetas/720531-pimientos-rellenos-de-crema-de-atun.

Apéndices:

1. *Script SQL – "Create tables"*

```
DROP TABLE IF EXISTS `ing_fridge`;
DROP TABLE IF EXISTS `ing_rec`;
DROP TABLE IF EXISTS `ingredients`;
DROP TABLE IF EXISTS `recipes`;

CREATE TABLE `recipes` (
`id` INT(6) unsigned auto_increment,
`title` VARCHAR(30) NOT NULL,
`description` VARCHAR(1255) NOT NULL,
PRIMARY KEY(`id`)
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `ingredients` (
`id` INT(6) unsigned auto_increment,
`name` VARCHAR(30) NOT NULL,
PRIMARY KEY(`id`)
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `ing_rec` (
`id` INT(6) unsigned auto_increment,
`r_id` INT(6) unsigned,
`i_id` INT(6) unsigned,
PRIMARY KEY(`id`),
FOREIGN KEY (`r_id`) REFERENCES recipes (`id`),
FOREIGN KEY (`i_id`) REFERENCES ingredients (`id`)
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `ing_fridge` (
`id` INT(6) unsigned auto_increment,
`i_id` INT(6) unsigned,
`quantity` INT(6),
PRIMARY KEY(`id`),
FOREIGN KEY (`i_id`) REFERENCES ingredients (`id`)
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

INSERT INTO `ingredients` (name) VALUES ('patata');
INSERT INTO `ingredients` (name) VALUES ('huevo');
INSERT INTO `ingredients` (name) VALUES ('tomate');
INSERT INTO `ingredients` (name) VALUES ('plátano');
INSERT INTO `ingredients` (name) VALUES ('limón');
INSERT INTO `ingredients` (name) VALUES ('pimiento');
INSERT INTO `ingredients` (name) VALUES ('manzana');
INSERT INTO `ingredients` (name) VALUES ('naranja');
INSERT INTO `ingredients` (name) VALUES ('ajo');
INSERT INTO `ingredients` (name) VALUES ('cebolla');
INSERT INTO `ingredients` (name) VALUES ('espárrago triguero');
INSERT INTO `ingredients` (name) VALUES ('atún');
INSERT INTO `ingredients` (name) VALUES ('jamón');
INSERT INTO `ingredients` (name) VALUES ('queso');
INSERT INTO `ingredients` (name) VALUES ('pulpo');



INSERT INTO `recipes` (title,description) VALUES
('Revuelto_de_tomate','REVUELTO DE TOMATE
```

```
Tiempo ~10 min

Ingredientes necesarios para 1 persona: 2 tomates, 1 huevo, sal.

Paso 1: Rallar los tomates y freirlos.
Paso 2: Añadir el huevo y remover lentamente hasta que cuaje,
salando al gusto.');
INSERT INTO `recipes` (title,description) VALUES
('Pulpo_a_la_gallega','PULPO A LA GALLEGA

Tiempo ~15 min

Ingredientes necesarios para 2 personas: 2 patas de pulpo cocido,
4 patatas pequeñas, sal, pimentón dulce.

Paso 1: Cocer las patatas con sal. Partirlas en rodajas y ponerlas
en el plato.
Paso 2: Lavar y cortar el pulpo, poniéndolo sobre las patatas.
Paso 3: Añadirle aceite, sal y un poco de pimenton dulce
Paso 4: Ponerlo 5 minutos al microondas.');
INSERT INTO `recipes` (title,description) VALUES
('Tortilla_de_patatas','TORTILLA DE PATATAS

Tiempo ~10 min

Ingredientes necesarios para 2 personas: 350g patatas, 3 huevos,
sal.

Paso 1: Pelar y cortar las patatas en láminas finas.
Paso 2: Salar al gusto y las cocinamos en el microondas 5 minutos
a 800W.
Paso 3: Mientras se hacen, batir los huevos y ponemos a calentar
la sartén
Paso 4: Mezclar las patatas con el huevo y la vertemos en la
sartén');
INSERT INTO `recipes` (title,description) VALUES
('Plátanos_Fritos','PLATANOS FRITOS

Tiempo ~10 min

Ingredientes necesarios para 2 personas: 3 plátanos, 1 huevo, 45g
Harina y 50 ml de Agua.

Paso 1: Pelar y cortamos los plátanos en 4.
Paso 2: Rebozar los trozos en huevo y harina
Paso 3: Freír hasta que se doren por los dos lados');
INSERT INTO `recipes` (title,description) VALUES
('Pimientos_rellenos_de_atún','PIMIENTOS RELLENOS DE ATÚN

Tiempo ~20 min

Ingredientes necesarios para 2 personas: 8 pimientos, 2 latas de
atún, 1 tomate.

Paso 1: Triturar el atún, rallar el tomate y mezclarlo.
Paso 2: Rellenar los pimientos con la pasta de atún.
Paso 3: Hornear 10 minutos a 180°');

INSERT INTO `ing_rec` (r_id,i_id) VALUES (1,2);
```

```
INSERT INTO `ing_rec` (r_id,i_id) VALUES (1,3);

INSERT INTO `ing_rec` (r_id,i_id) VALUES (2,1);
INSERT INTO `ing_rec` (r_id,i_id) VALUES (2,15);

INSERT INTO `ing_rec` (r_id,i_id) VALUES (3,1);
INSERT INTO `ing_rec` (r_id,i_id) VALUES (3,2);

INSERT INTO `ing_rec` (r_id,i_id) VALUES (4,2);
INSERT INTO `ing_rec` (r_id,i_id) VALUES (4,4);

INSERT INTO `ing_rec` (r_id,i_id) VALUES (5,3);
INSERT INTO `ing_rec` (r_id,i_id) VALUES (5,6);
INSERT INTO `ing_rec` (r_id,i_id) VALUES (5,12);
```

2. *Ingredientes*
    1. patata
    2. huevo
    3. tomate
    4. plátano
    5. limón
    6. pimiento
    7. manzana
    8. naranja
    9. ajo
    10. cebolla
    11. espárrago triguero
    12. atún
    13. jamón
    14. queso
    15. pulpo

Entrenados en la red Neuronal

3. *Recetas*
    1. Revuelto de tomate [9] [10]
    2. Pulpo a la gallega [11]
    3. Tortilla de patatas [12]
    4. Plátanos Fritos [13]
    5. Pimientos rellenos de atún [14]

4. *Móviles*
    1. iphone 5s
    2. iphone 6s
    3. iphone 6
    4. samsung galaxy grand prime
    5. nokia
    6. Sony xperia m4 aqua
5. *Scripts de Matlab*

Índice

```
function varargout = Fast_Cooking(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fast_Cooking_OpeningFcn, ...
                   'gui_OutputFcn',  @Fast_Cooking_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fast Cooking is made visible.
function Fast_Cooking_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Fast_Cooking (see VARARGIN)

% Choose default command line output for Fast_Cooking
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fast_Cooking wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Logo

[a,b,c]=imread('logo.jpg');
axes(handles.logo)
imshow(a);

%Install & Compile Matconvnet
cd('C:\Users\Alberto\Documents\MATLAB\TFG\matconvnet\matlab')
vl_setupnn
vl_compilenn


% --- Outputs from this function are returned to the command line.
function varargout = Fast_Cooking_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Guardar_Alimentos.
function Guardar_Alimentos_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar_Alimentos (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Fast_Cooking1
delete(handles.figure1)

% --- Executes on button press in Mi_Nevera.
function Mi_Nevera_Callback(hObject, eventdata, handles)
% hObject    handle to Mi_Nevera (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking2
delete(handles.figure1)

% --- Executes on button press in Recetas.
function Recetas_Callback(hObject, eventdata, handles)
% hObject    handle to Recetas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking3
delete(handles.figure1)

% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
% hObject    handle to Salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%global conn;
%close(conn);
close
```

Guardar Alimento

```
function varargout = Fast_Cooking1(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fast_Cooking1_OpeningFcn, ...
                   'gui_OutputFcn',  @Fast_Cooking1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fast_Cooking1 is made visible.
function Fast_Cooking1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to Fast_Cooking1 (see VARARGIN)

% Choose default command line output for Fast_Cooking1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fast_Cooking1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Ocultar Axes

set(handles.Foto_Al,'visible','off');

%Logo

[a,b,c]=imread('logo.jpg');
axes(handles.logo)
imshow(a);


% --- Outputs from this function are returned to the command line.
function varargout = Fast_Cooking1_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking
delete(handles.figure1)


% --- Executes on button press in Mi_Nevera.
function Mi_Nevera_Callback(hObject, eventdata, handles)
% hObject    handle to Mi_Nevera (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking2
delete(handles.figure1)

% --- Executes on button press in Limpiar.
function Limpiar_Callback(hObject, eventdata, handles)
% hObject    handle to Limpiar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.Foto_Al)
set(handles.Alimento,'string','')
set(handles.Cantidad,'string','')
```

```
% --- Executes on button press in Guardar.
function Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


if strcmp(get(handles.Alimento,'string'),'')==1 |
strcmp(get(handles.Cantidad,'string'),'')==1

else

    Load_DB;
    ing=get(handles.Alimento,'string');
    q=str2double(get(handles.Cantidad,'string'));
    for i=1:length(ingredients.id)
        cmp=strcmp(ing,ingredients.name(i));
        if cmp == 1
            if strcmp(ing_fridge{1,1},'No Data')==0
                for j=1:length(ing_fridge.id)
                   if ing_fridge.i_id(j) ==i
                       q2=q+ing_fridge.quantity(j);
                       DB_Update1(conn,i,q2)
                       break
                   else
                       if j==length(ing_fridge.id)
                           DB_Insert_F(conn,i,q)
                           break
                       end
                   end
                end
                break
            else
                DB_Insert_F(conn,i,q)
                break
            end

        end
    end
    if cmp~=1
        %if strcmp(ing_fridge{1,1},'No Data')==0
            DB_Insert1(conn,length(ingredients.id)+1,ing,q)
        %else
            %DB_Insert1(conn,length(ingredients.id)+1,ing,q)
        %end
    end
    close(conn);
    %Clean all
    delete(handles.Foto_Al)
    set(handles.Alimento,'string','')
    set(handles.Cantidad,'string','')
end

function Cantidad_Callback(hObject, eventdata, handles)
% hObject    handle to Cantidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Cantidad as text
```

```
%        str2double(get(hObject,'String')) returns contents of Cantidad
as a double


% --- Executes during object creation, after setting all properties.
function Cantidad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Cantidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Alimento_Callback(hObject, eventdata, handles)
% hObject    handle to Alimento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Alimento as text
%        str2double(get(hObject,'String')) returns contents of Alimento
as a double


% --- Executes during object creation, after setting all properties.
function Alimento_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Alimento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Selec_Foto.
function Selec_Foto_Callback(hObject, eventdata, handles)
% hObject    handle to Selec_Foto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
path='C:\Users\Alberto\Documents\MATLAB\TFG\Examples';
cd(path)
%cd('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimentos\originales
\internet')
[filename pathname] = uigetfile({'*.jpg';'*.bmp';'*.png'},'File
Selector');
image = strcat(pathname, filename);
%if strcmp(path,pathname)~=1
    %copyfile(image,'C:\Users\Alberto\Documents\MATLAB\TFG\Examples')
%end
```

```
fid=fopen('task.txt','w');
fprintf(fid,'%s \n',filename);
res=extract_features();
cd('C:\Users\Alberto\Documents\MATLAB\TFG')


set(handles.Alimento, 'string', res);


%Show user's image
axes(handles.Foto_Al)
imshow(image)


% --- Executes on button press in Recetas.
function Recetas_Callback(hObject, eventdata, handles)
% hObject    handle to Recetas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking3
delete(handles.figure1)


% --- Executes on mouse press over axes background.
function logo_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to logo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Scripts Database

```
%Connect Database

function conn=connect_DB()
%Set preferences with setdbprefs.
setdbprefs('DataReturnFormat', 'table');
setdbprefs('NullNumberRead', 'NaN');
setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
conn = database('Fast_Cooking_DB', 'root', 'rootroot');


%Select tables
function ing_fridge=ing_fridge_DB(conn)
%Set preferences with setdbprefs.
setdbprefs('DataReturnFormat', 'table');
setdbprefs('NullNumberRead', 'NaN');
setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
%conn = database('Fast_Cooking_DB', 'root', '');

%Read data from database.
curs = exec(conn, ['SELECT  ing_fridge.id'...
```

```
    ' , ing_fridge.i_id'...
    ' , ing_fridge.quantity'...
    ' FROM  fast_cooking_db.ing_fridge ']);

curs = fetch(curs);
close(curs);

%Assign data to output variable
ing_fridge = curs.Data;


function ing_rec=ing_rec_DB(conn)
%Set preferences with setdbprefs.
setdbprefs('DataReturnFormat', 'table');
setdbprefs('NullNumberRead', 'NaN');
setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
%conn = database('Fast_Cooking_DB', 'root', '');

%Read data from database.
curs = exec(conn, ['SELECT  ing_rec.id'...
    ' , ing_rec.r_id'...
    ' , ing_rec.i_id'...
    ' FROM  fast_cooking_db.ing_rec ']);

curs = fetch(curs);
close(curs);

%Assign data to output variable
ing_rec = curs.Data;


function ingredients=ingredients_DB(conn)
%Set preferences with setdbprefs.
setdbprefs('DataReturnFormat', 'table');
setdbprefs('NullNumberRead', 'NaN');
setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
%conn = database('Fast_Cooking_DB', 'root', '');

%Read data from database.
curs = exec(conn, ['SELECT  ingredients.id'...
    ' , ingredients.name'...
    ' FROM  fast_cooking_db.ingredients ']);

curs = fetch(curs);
close(curs);

%Assign data to output variable
ingredients = curs.Data;


function Nevera=Nevera_DB(conn)
%Set preferences with setdbprefs.
```

```
%setdbprefs('DataReturnFormat', 'table');
%setdbprefs('NullNumberRead', 'NaN');
%setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
%conn = database('Fast_Cooking_DB', 'root', '');

%Read data from database.
curs = exec(conn, ['SELECT  nevera.id'...
    ' , nevera.Alimento'...
    ' , nevera.Cantidad'...
    ' FROM  fast_cooking_db.nevera ']);

curs = fetch(curs);
close(curs);

%Assign data to output variable
Nevera = curs.Data;

function recipes=recipes_DB(conn)
%Set preferences with setdbprefs.
setdbprefs('DataReturnFormat', 'table');
setdbprefs('NullNumberRead', 'NaN');
setdbprefs('NullStringRead', 'null');


%Make connection to database.  Note that the password has been omitted.
%Using ODBC driver.
%conn = database('Fast_Cooking_DB', 'root', '');

%Read data from database.
curs = exec(conn, ['SELECT  recipes.id'...
    ' , recipes.title'...
    ' , recipes.description'...
    ' FROM  fast_cooking_db.recipes ']);

curs = fetch(curs);
close(curs);

%Assign data to output variable
recipes = curs.Data;

%Close database connection.
%close(conn);

%Clear variables
%clear curs conn


%Load Database

global conn;
global ing_fridge;
global ing_rec;
global ingredients;
global recipes;
conn=connect_DB;
ing_fridge=ing_fridge_DB(conn);
```

```
ing_rec=ing_rec_DB(conn);
ingredients=ingredients_DB(conn);
recipes=recipes_DB(conn);


%Insert
function DB_Insert_F(conn,id,q)
colnames={'i_id','quantity'};
data={id,q};
tablename='ing_fridge';
data_table = cell2table(data,'VariableNames',colnames);
insert(conn,tablename,colnames,data_table)

function DB_Insert1(conn,id,ing,q)
colnames = {'id','name'};
colnames2={'i_id','quantity'};
data={id,ing};
data2={id,q};
tablename = 'ingredients';
tablename2='ing_fridge';
data_table = cell2table(data,'VariableNames',colnames);
data_table2 = cell2table(data2,'VariableNames',colnames2);
insert(conn,tablename,colnames,data_table)
insert(conn,tablename2,colnames2,data_table2)

%Update

function DB_Update1(conn,id,q)
colnames = {'quantity'};
data={q};
tablename = 'ing_fridge';
whereclause = {sprintf('where i_id =%d',id)};
update(conn,tablename,colnames,data,whereclause)
```

Evaluación imagen

```
function res=extract_features()

cd('C:\Users\Alberto\Documents\MATLAB\TFG\matconvnet\matlab')

%% input files
root_path = 'C:\Users\Alberto\Documents\MATLAB\TFG\Examples\';
tasks = [root_path 'task.txt'];
fs = textread(tasks, '%s');

%% MatConvNet network
cd('C:\Users\Alberto\Documents\MATLAB\TFG\matconvnet\scripts')
model_file = 'bueno1';
net = load(model_file);

%% normalization
im = [root_path,fs{1}];
im=imread(im);
im=imresize(im,[32 32]);
im_ = bsxfun(@minus, single(im), net.mean);
%contrast

z = reshape(im_,[],1) ;
z = bsxfun(@minus, z, mean(z,1)) ;
```

```
n = std(z,0,1) ;
z = bsxfun(@times, z, mean(n) ./ max(n, 40)) ;
im_2 = reshape(z, 32, 32, 3, []) ;

%whiten

z = reshape(im_2,[],1) ;
W = z(:)*z(:)' ;
[V,D] = eig(W) ;
% the scale is selected to approximately preserve the norm of W
d2 = diag(D) ;
en = sqrt(mean(d2)) ;
z = V*diag(en./max(sqrt(d2), 10))*V'*z ;
im_3 = reshape(z, 32, 32, 3, []) ;

%% feats simplenn
cd('C:\Users\Alberto\Documents\MATLAB\TFG\matconvnet\matlab\simplenn')
net=vl_simplenn_tidy(net.net);
net.layers{end}.type = 'softmax';
out=[];
dzdy = [] ;
evalMode = 'test' ;
out= vl_simplenn(net, im_2, dzdy, out, ...
                         'mode', evalMode);
output = out(end).x;
pos=find(output==max(output));
cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
res=ali(pos);
```

Mi Nevera

```
function varargout = Fast_Cooking2(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fast_Cooking2_OpeningFcn, ...
                   'gui_OutputFcn',  @Fast_Cooking2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fast_Cooking1 is made visible.
function Fast_Cooking2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Fast_Cooking2 (see VARARGIN)
```

```
% Choose default command line output for Fast_Cooking2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fast_Cooking2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Logo

[a,b,c]=imread('logo.jpg');
axes(handles.logo)
imshow(a);

%Load Database

Load_DB;

%Load listbox
if strcmp(ing_fridge{1,1},'No Data')==0
Load_Listbox;
end

close(conn);

% --- Outputs from this function are returned to the command line.
function varargout = Fast_Cooking2_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking
delete(handles.figure1)


% --- Executes on selection change in Alimentos.
function Alimentos_Callback(hObject, eventdata, handles)
% hObject    handle to Alimentos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Alimentos
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
Alimentos
```

```
% --- Executes during object creation, after setting all properties.
function Alimentos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Alimentos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Quitar.
function Quitar_Callback(hObject, eventdata, handles)
% hObject    handle to Quitar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Load_DB;

if strcmp(ing_fridge{1,1},'No Data')==0
    index=get(handles.Alimentos, 'Value');
    id=ing_fridge.i_id(index);
    ing_fridge.quantity(index)=0;
    Load_Listbox;
    DB_Update1(conn,id,0)
end
close(conn);


% --- Executes on button press in Modificar.
function Modificar_Callback(hObject, eventdata, handles)
% hObject    handle to Modificar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Load_DB;

if strcmp(ing_fridge{1,1},'No Data')==0

    index=get(handles.Alimentos, 'Value');
    id=ing_fridge.i_id(index);

ing_fridge.quantity(index)=str2double(get(handles.Cantidad,'string'));
    Load_Listbox;
    DB_Update1(conn,id,ing_fridge.quantity(index))
end
close(conn);

function Cantidad_Callback(hObject, eventdata, handles)
% hObject    handle to Cantidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Cantidad as text
%        str2double(get(hObject,'String')) returns contents of Cantidad
as a double
```

```
% --- Executes during object creation, after setting all properties.
function Cantidad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Cantidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Guardar_Alimento.
function Guardar_Alimento_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar_Alimento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking1
delete(handles.figure1)


% --- Executes on button press in Recetas.
function Recetas_Callback(hObject, eventdata, handles)
% hObject    handle to Recetas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking3
delete(handles.figure1)


% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Load Listbox y Popup

```
% Load Listbox

for i=1:length(ing_fridge.id)
    Lista(i)=strcat({'  '},ingredients.name(ing_fridge.i_id(i)),{'  ----
->  '},int2str(ing_fridge.quantity(i)));
end
set(handles.Alimentos, 'String', Lista');


% Load popup

sql_qdisp=['SELECT  ing_fridge.i_id FROM    fast_cooking_db.ing_fridge
WHERE ing_fridge.quantity!=0'];
setdbprefs('DataReturnFormat', 'cellArray');
curs = exec(conn, sql_qdisp);
curs = fetch(curs);
close(curs);
```

```
ing_disp = curs.Data;
z=2;%Id de la lista
Lista(1)={'Recetas Disponibles'};
for i=1:length(recipes.id)
    %Coger id de los ingredientes necesarios para cada receta
    sql_i_id=sprintf('SELECT ing_rec.i_id FROM fast_cooking_db.ing_rec,
fast_cooking_db.recipes WHERE recipes.id=ing_rec.r_id AND
recipes.id=%d',i);
    setdbprefs('DataReturnFormat', 'cellArray');
    curs = exec(conn, sql_i_id);
    curs = fetch(curs);
    close(curs);
    ing_n = curs.Data;
    for k=1:length(ing_n)
        for l=1:length(ing_disp)
            res=ing_n{k}==ing_disp{l};
            if res==1
                break
            end
        end
        if res==0
            break
        end
    end
    if res == 1
        Lista(z)=recipes.title(i);
        z=z+1;
    end
end

set(handles.Recetas, 'String', Lista);
```

Recetas

```
function varargout = Fast_Cooking3(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fast_Cooking3_OpeningFcn, ...
                   'gui_OutputFcn',  @Fast_Cooking3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fast_Cooking1 is made visible.
function Fast_Cooking3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Fast_Cooking3 (see VARARGIN)

% Choose default command line output for Fast_Cooking3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fast_Cooking1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

set(handles.Foto_Receta,'visible','off');
set(handles.Descripcion,'visible','off');
%Logo

[a,b,c]=imread('logo.jpg');
axes(handles.logo)
imshow(a);

%Load Database and popup

Load_DB;
Load_popup;



% --- Outputs from this function are returned to the command line.
function varargout = Fast_Cooking3_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking
delete(handles.figure1)


% --- Executes on button press in Guardar_Alimento.
function Guardar_Alimento_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar_Alimento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking1
delete(handles.figure1)

% --- Executes on button press in Nevera.
function Nevera_Callback(hObject, eventdata, handles)
% hObject    handle to Nevera (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fast_Cooking2
delete(handles.figure1)


% --- Executes on selection change in Recetas.
function Recetas_Callback(hObject, eventdata, handles)
% hObject    handle to Recetas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Recetas
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
Recetas

%Load Recipe
set(handles.Foto_Receta,'visible','off');
set(handles.Descripcion,'visible','off');
Load_DB;
all_names=get(handles.Recetas, 'String');
id=get(handles.Recetas, 'Value');
if id ~= 1
    set(handles.Foto_Receta,'visible','on');
    set(handles.Descripcion,'visible','on');
    name=all_names{id};
    name2=sprintf('''%s''',name);
    sql_rec=sprintf('SELECT recipes.id FROM fast_cooking_db.recipes
WHERE recipes.title = %s',name2);
    setdbprefs('DataReturnFormat', 'cellArray');
    curs = exec(conn, sql_rec);
    curs = fetch(curs);
    close(curs);
    index = curs.Data;
    descr=recipes.description(cell2mat(index));
    set(handles.Descripcion, 'String', descr);
    pathname =
'C:\Users\Alberto\Documents\MATLAB\TFG\Database\Recetas\';
    filename=strcat(name,'.jpg');
    image = strcat(pathname,filename);
    %Show user's image
    axes(handles.Foto_Receta)
    imshow(image)
end

% --- Executes during object creation, after setting all properties.
function Recetas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Recetas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Descripcion_Callback(hObject, eventdata, handles)
% hObject    handle to Descripcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Descripcion as text
%        str2double(get(hObject,'String')) returns contents of
Descripcion as a double


% --- Executes during object creation, after setting all properties.
function Descripcion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Descripcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Entrenamiento Red Neuronal

```
function [net, info] = cnn_fastcooking(varargin)

run(fullfile(fileparts(mfilename('fullpath')), ...
  '..', '..', 'matlab', 'vl_setupnn.m')) ;

opts.modelType = 'lenet' ;
[opts, varargin] = vl_argparse(opts, varargin) ;

opts.expDir = fullfile(vl_rootnn, 'data', ...
  sprintf('fast_cooking-%s', opts.modelType)) ;
[opts, varargin] = vl_argparse(opts, varargin) ;

opts.dataDir = fullfile(vl_rootnn, 'data','fast_cooking') ;
opts.imdbPath = fullfile(opts.expDir, 'imdb.mat');
opts.whitenData = true ;
opts.contrastNormalization = true ;
opts.networkType = 'simplenn' ;
opts.train = struct() ;
opts = vl_argparse(opts, varargin) ;
if ~isfield(opts.train, 'gpus'), opts.train.gpus = []; end;

% -------------------------------------------------------------------
---
%                                                  Prepare model and
data
% -------------------------------------------------------------------
---

switch opts.modelType
  case 'lenet'
    net = cnn_FastCooking_init('networkType', opts.networkType) ;
```

```
  otherwise
    error('Unknown model type ''%s''.', opts.modelType) ;
end

if exist(opts.imdbPath, 'file')
  imdb = load(opts.imdbPath) ;
else
  imdb = getFastCookingImdb(opts) ;
  mkdir(opts.expDir) ;
  save(opts.imdbPath, '-struct', 'imdb') ;
end

net.meta.classes.name = imdb.meta.classes(:)' ;

% -------------------------------------------------------------------------
---
%
Train
% -------------------------------------------------------------------------
---

switch opts.networkType
  case 'simplenn', trainfn = @cnn_train ;
end

[net, info] = trainfn(net, imdb, getBatch(opts), ...
  'expDir', opts.expDir, ...
  net.meta.trainOpts, ...
  opts.train, ...
  'val', find(imdb.images.set == 3)) ;

% -------------------------------------------------------------------------
---
function fn = getBatch(opts)
% -------------------------------------------------------------------------
---
switch lower(opts.networkType)
  case 'simplenn'
    fn = @(x,y) getSimpleNNBatch(x,y) ;
end

% -------------------------------------------------------------------------
---
function [images, labels] = getSimpleNNBatch(imdb, batch)
% -------------------------------------------------------------------------
---
images = imdb.images.data(:,:,:,batch) ;
labels = imdb.images.labels(1,batch) ;
if rand > 0.5, images=fliplr(images) ; end

% -------------------------------------------------------------------------
---
function imdb = getFastCookingImdb(opts)
% -------------------------------------------------------------------------
---
% Preapre the imdb structure, returns image data with mean image
subtracted
unpackPath = fullfile(opts.dataDir, 'fastcooking-10-batches-mat');
```

```
files = [arrayfun(@(n) sprintf('data_batch_%d.mat', n), 1:5,
'UniformOutput', false) ...
  {'test_batch.mat'}];
files = cellfun(@(fn) fullfile(unpackPath, fn), files, 'UniformOutput',
false);
file_set = uint8([ones(1, 5), 3]);

data = cell(1, numel(files));
labels = cell(1, numel(files));
sets = cell(1, numel(files));
for fi = 1:numel(files)
  fd = load(files{fi}) ;
  data{fi} = permute(reshape(fd.data',32,32,3,[]),[2 1 3 4]) ;
  labels{fi} = fd.labels' + 1; % Index from 1
  sets{fi} = repmat(file_set(fi), size(labels{fi}));
end

set = cat(2, sets{:});
data = single(cat(4, data{:}));

% remove mean in any case
dataMean = mean(data(:,:,:,set == 1), 4);
data = bsxfun(@minus, data, dataMean);
% normalize by image mean and std as suggested in `An Analysis of
% Single-Layer Networks in Unsupervised Feature Learning` Adam
% Coates, Honglak Lee, Andrew Y. Ng

if opts.contrastNormalization
  z = reshape(data,[],60000) ;
  z = bsxfun(@minus, z, mean(z,1)) ;
  n = std(z,0,1) ;
  z = bsxfun(@times, z, mean(n) ./ max(n, 40)) ;
  data = reshape(z, 32, 32, 3, []) ;
end

if opts.whitenData
  z = reshape(data,[],60000) ;
  W = z(:,set == 1)*z(:,set == 1)'/60000 ;
  [V,D] = eig(W) ;
  % the scale is selected to approximately preserve the norm of W
  d2 = diag(D) ;
  en = sqrt(mean(d2)) ;
  z = V*diag(en./max(sqrt(d2), 10))*V'*z ;
  data = reshape(z, 32, 32, 3, []) ;
end

clNames = load(fullfile(unpackPath, 'batches.meta.mat'));

imdb.images.data = data ;
imdb.images.labels = single(cat(2, labels{:})) ;
imdb.images.set = set;
imdb.meta.sets = {'train', 'val', 'test'} ;
imdb.meta.classes = clNames.label_names;
imdb.mean = dataMean;


function net = cnn_FastCooking_init(varargin)
opts.networkType = 'simplenn' ;
opts = vl_argparse(opts, varargin) ;
```

```
lr = [.1 2] ;

% Define network Fast Cooking
net.layers = {} ;

%% Block 1
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.01*randn(5,5,3,32, 'single'),
zeros(1, 32, 'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;
net.layers{end+1} = struct('type', 'relu') ;
% % add dropout layers in network (saved model has dropout removed)
net.layers{end+1} = struct('type', 'dropout', 'rate' , 0.3) ;
% Block 2
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(5,5,32,32, 'single'),
zeros(1,32,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'relu') ;
% % add dropout layers in network (saved model has dropout removed)
net.layers{end+1} = struct('type', 'dropout', 'rate' , 0.3) ;
net.layers{end+1} = struct('type', 'pool', ...
                            'method', 'avg', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ; % Emulate caffe


% Block 3
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(5,5,32,64, 'single'),
zeros(1,64,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'relu') ;
% % add dropout layers in network (saved model has dropout removed)
%net.layers{end+1} = struct('type', 'dropout', 'rate' , 0.3) ;

net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ; % Emulate caffe


% Block 4
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(4,4,64,64, 'single'),
zeros(1,64,'single')}}, ...
                           'learningRate', lr, ...
```

```
                            'stride', 1, ...
                            'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;


% Block 5
net.layers{end+1} = struct('type', 'conv', ...
                            'weights', {{0.05*randn(1,1,64,10, 'single'),
zeros(1,10,'single')}}, ...
                            'learningRate', .1*lr, ...
                            'stride', 1, ...
                            'pad', 0) ;

% Loss layer
net.layers{end+1} = struct('type', 'softmaxloss') ;



%Meta parameters
net.meta.inputSize = [32 32 3] ;
net.meta.trainOpts.learningRate = [0.01*ones(1,30) 0.001*ones(1,10)
0.0001*ones(1,5)] ;
net.meta.trainOpts.weightDecay = 0.0001 ;
net.meta.trainOpts.batchSize = 128 ;
net.meta.trainOpts.numEpochs = numel(net.meta.trainOpts.learningRate) ;

% Fill in default values
net = vl_simplenn_tidy(net) ;
```

Dataset – Normalización y guardado

```
function resize_pre()
cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
for z=1:length(ali)

folder_in=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Aliment
os\originales\train\',ali{z},'\');

folder_in2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\originales\test\',ali{z},'\');

folder_out=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\resize_128\train\',ali{z},'\');

folder_out2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alime
ntos\resize_128\test\',ali{z},'\');
    read_ims = dir([folder_in,'*.JPG']);
    read_ims2 = dir([folder_in2,'*.JPG']);

    for i = 1:length(read_ims)
        cd(folder_in)
        im = read_ims(i).name;
        im=imread(im);
        im=imresize(im,[128 128]);
        out=[ali{z},int2str(i),'.jpg'];
```

```
        cd(folder_out)
        imwrite(im,out)
    end
     for j = 1:length(read_ims2)
        cd(folder_in2)
        im = read_ims2(j).name;
        im=imread(im);
        im=imresize(im,[128 128]);
        out=[ali{z},int2str(j),'.jpg'];
        cd(folder_out2)
        imwrite(im,out)
    end
end
cd('C:\Users\Alberto\Documents\MATLAB\TFG')


function resize_post()
cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
for z=1:length(ali)

folder_in=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Aliment
os\resize_128\train\',ali{z},'\');

folder_in2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\resize_128\test\',ali{z},'\');

folder_out=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\resize_32\train\',ali{z},'\');

folder_out2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alime
ntos\resize_32\test\',ali{z},'\');
    read_ims = dir([folder_in,'*.JPG']);
    read_ims_da = dir([folder_in,'*.png']);
    read_ims2 = dir([folder_in2,'*.JPG']);
    read_ims_da2 = dir([folder_in2,'*.png']);

    for i = 1:length(read_ims)
        cd(folder_in)
        im = read_ims(i).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        out=[ali{z},int2str(i),'.jpg'];
        cd(folder_out)
        imwrite(im,out)

    end
    for j = 1:length(read_ims_da)
        cd(folder_in)
        im = read_ims_da(j).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        out=[ali{z},int2str(length(read_ims)+j),'.jpg'];
        cd(folder_out)
        imwrite(im,out)

    end
    for k = 1:length(read_ims2)
        cd(folder_in2)
        im = read_ims2(k).name;
```

```
        im=imread(im);
        im=imresize(im,[32 32]);
        out=[ali{z},int2str(k),'.jpg'];
        cd(folder_out2)
        imwrite(im,out)

    end
    for l = 1:length(read_ims_da2)
        cd(folder_in2)
        im = read_ims_da2(l).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        out=[ali{z},int2str(length(read_ims2)+l),'.jpg'];
        cd(folder_out2)
        imwrite(im,out)

    end
end
cd('C:\Users\Alberto\Documents\MATLAB\TFG')


function resize_post_norm()
cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
for z=1:length(ali)

folder_in=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Aliment
os\resize_32\train\',ali{z},'\');

folder_in2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\resize_32\test\',ali{z},'\');

folder_out=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alimen
tos\Final\train\',ali{z},'\');

folder_out2=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alime
ntos\Final\test\',ali{z},'\');
    read_ims = dir([folder_in,'*.JPG']);
    read_ims_da = dir([folder_in,'*.png']);
    read_ims2 = dir([folder_in2,'*.JPG']);
    read_ims_da2 = dir([folder_in2,'*.png']);

    for i = 1:length(read_ims)
        cd(folder_in)
        im = read_ims(i).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        im=single(im);
        dataMean = mean(im(:,:,:), 3);
        im = bsxfun(@minus, im, dataMean);
        out=[ali{z},int2str(i),'.jpg'];
        cd(folder_out)
        imwrite(im,out)

    end
    for j = 1:length(read_ims_da)
        cd(folder_in)
        im = read_ims_da(j).name;
        im=imread(im);
        im=imresize(im,[32 32]);
```

```
        im=single(im);
        dataMean = mean(im(:,:,:), 3);
        im = bsxfun(@minus, im, dataMean);
        out=[ali{z},int2str(length(read_ims)+j),'.jpg'];
        cd(folder_out)
        imwrite(im,out)

    end
    for k = 1:length(read_ims2)
        cd(folder_in2)
        im = read_ims2(k).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        im=single(im);
        dataMean = mean(im(:,:,:), 3);
        im = bsxfun(@minus, im, dataMean);
        out=[ali{z},int2str(k),'.jpg'];
        cd(folder_out2)
        imwrite(im,out)

    end
    for l = 1:length(read_ims_da2)
        cd(folder_in2)
        im = read_ims_da2(l).name;
        im=imread(im);
        im=imresize(im,[32 32]);
        im=single(im);
        dataMean = mean(im(:,:,:), 3);
        im = bsxfun(@minus, im, dataMean);
        out=[ali{z},int2str(length(read_ims2)+l),'.jpg'];
        cd(folder_out2)
        imwrite(im,out)

    end
end
cd('C:\Users\Alberto\Documents\MATLAB\TFG')


function label_names()
cd('C:\Users\Alberto\Documents\MATLAB\TFG')
label_names=textread('alimentos.txt','%s');
folder_out='C:\Users\Alberto\Documents\MATLAB\TFG\Database\dataset';
folder_out2='C:\Users\Alberto\Documents\MATLAB\TFG\MatConvNet\data\fast_
cooking\fast_cooking50+10k';
cd(folder_out)
save('batches.meta.mat','label_names');
cd(folder_out2)
save('batches.meta.mat','label_names');
cd('C:\Users\Alberto\Documents\MATLAB\TFG')


function Save_dataset_train3()

cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
pos=0;
long=0;
start=1;
for z=1:2
```

```
folder_train=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alim
entos\Final\train\',ali{z+pos},'\');
    folder_out='C:\Users\Alberto\Documents\MATLAB\TFG\Database\dataset';

folder_out2='C:\Users\Alberto\Documents\MATLAB\TFG\MatConvNet\data\fastc
ooking\fastcooking-10-batches-mat';
    read_ims_train = dir([folder_train,'*.jpg']);
    %data
    for i = 1:length(read_ims_train)
        cd(folder_train)
        im = read_ims_train(i).name;
        im=imread(im);

        out1=im(:,:,1);
        out2=im(:,:,2);
        out3=im(:,:,3);
        out1=transpose(out1);
        out2=transpose(out2);
        out3=transpose(out3);
        out1=reshape(out1,1,1024);
        out2=reshape(out2,1,1024);
        out3=reshape(out3,1,1024);
        out4(i,:)=[out1,out2,out3];

    end
    if z==1
        data = out4;
    else
        data(5001:1:10000,:)=out4;
    end
    long=long+length(read_ims_train);
    %label


    for j=1:length(read_ims_train)
        if start==1
            labels=z-1+pos;
            start=0;
        else
            labels=[labels,z-1+pos];
        end
    end
end


labels=uint8(reshape(labels,long,1));
batch_label='training batch 1 of 5';
cd(folder_out)
save('data_batch_1.mat','batch_label','data','labels')
cd(folder_out2)
save('data_batch_1.mat','batch_label','data','labels')

function Save_dataset_test()

cd('C:\Users\Alberto\Documents\MATLAB\TFG')
ali=textread('alimentos.txt','%s');
long=0;
start=1;
for z=1:10
```

```
folder_test=strcat('C:\Users\Alberto\Documents\MATLAB\TFG\Database\Alime
ntos\Final\test\',ali{z},'\');
    folder_out='C:\Users\Alberto\Documents\MATLAB\TFG\Database\dataset';

folder_out2='C:\Users\Alberto\Documents\MATLAB\TFG\MatConvNet\data\fastc
ooking\fastcooking-10-batches-mat';
    read_ims_test = dir([folder_test,'*.jpg']);
    %data
    for i = 1:length(read_ims_test)
        cd(folder_test)
        im = read_ims_test(i).name;
        im=imread(im);

        out1=im(:,:,1);
        out2=im(:,:,2);
        out3=im(:,:,3);
        out1=transpose(out1);
        out2=transpose(out2);
        out3=transpose(out3);
        out1=reshape(out1,1,1024);
        out2=reshape(out2,1,1024);
        out3=reshape(out3,1,1024);
        out4(i,:)=[out1,out2,out3];

    end
    switch z
        case 1
            data = out4;
        case 2
            data(1001:1:2000,:)=out4;
        case 3
            data(2001:1:3000,:)=out4;
        case 4
            data(3001:1:4000,:)=out4;
        case 5
            data(4001:1:5000,:)=out4;
        case 6
            data(5001:1:6000,:)=out4;
        case 7
            data(6001:1:7000,:)=out4;
        case 8
            data(7001:1:8000,:)=out4;
        case 9
            data(8001:1:9000,:)=out4;
        case 10
            data(9001:1:10000,:)=out4;

    end


    long=long+length(read_ims_test);
    %label


    for j=1:length(read_ims_test)
        if start==1
            labels=z-1;
            start=0;
        else
```

```
            labels=[labels,z-1];
        end
    end
end


labels=uint8(reshape(labels,long,1));
batch_label='testing batch 1 of 1';
cd(folder_out)
save('test_batch.mat','batch_label','data','labels')
cd(folder_out2)
save('test_batch.mat','batch_label','data','labels')

cd('C:\Users\Alberto\Documents\MATLAB\TFG')
```